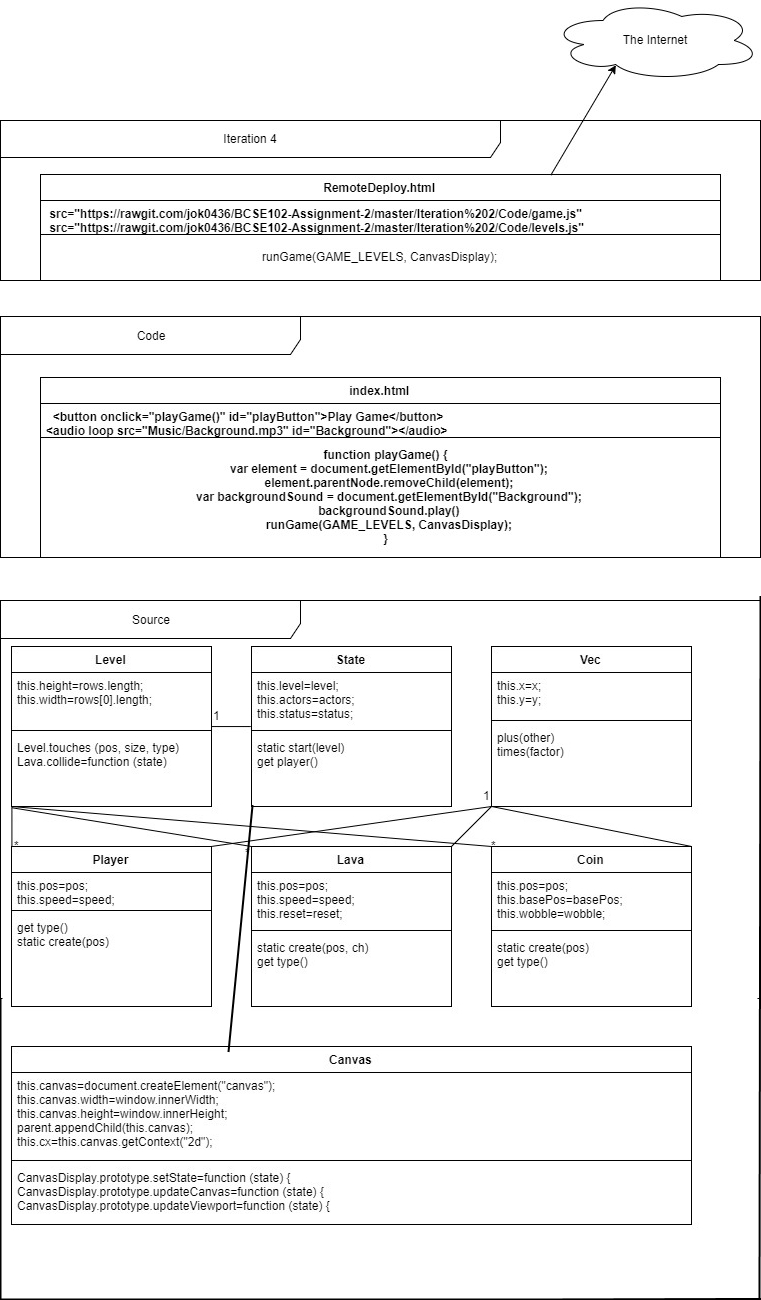
Iteration: 4

UML Diagram 1: Class Diagram of Current Code



UML Diagram 2: Package Diagram of Current Code

A screenshot of a cell phone

Description generated with high confidence

Iteration Work Plan:

* The Goal of The Iteration
  + Comment and Properly Separate Code
  + Lives
  + Background animation
  + A theme for the game relevant to the Christchurch rebuild. (a saying, a person, an identifiable place)
  + Uses images derived from the Christchurch Rebuild.
* The Planned Tasks in Sequence [Planning, Analysis, Design,

Coding, Testing]

* + Comment Code – A
  + Add Lives – B
  + Animate Background – C
  + Add Game Theme (Using Christchurch Rebuild Images) – D
* A Time Estimate for Each Task [30 Minute Blocks]
  + A – 120 Minutes
  + B – 60 Minutes
  + C – 30 Minutes
  + D - 30 Minutes
* The Planned ‘Product’ Of Each Task
  + Code is properly separated and commented - A
  + Player will lose the game if they run out of lives - B
  + Background is animated - C
  + Game has a Christchurch rebuild theme - D
* A Record of The Actual Time Each Task Took
  + A – 180 Minutes
  + B –
  + C – 120 Minutes
  + D –

**PLANNING A COMPLEX ALGORITHM**

**DESIGN THE ROUTINE**

CHECK PREREQUISITES

Define the problem

*Read a text file on the users computer*

Information the routine will hide

The levels will not be read as a javascript file

Inputs to the routine

*Levels.txt*

Outputs from the routine

The split levels as an [], split with the BREAK word

Pre-conditions

*The user has inputted a levels file*

Post-conditions

We can use the levels file to create all the levels

Name the Routine

*findGameLevels and getFile*

Decide how to test the routine

The levels file shows up in the console after a log

Research functionality available in standard libraries

*Need to use callback methods, specifically a javascript promise with the await keyword in an asynchronous function*

Think about error handling

Handled outside of this function

Think about efficiency

Javascript has an inbuilt class for reading files

**WRITE PSEUDOCODE**

Take in an input file event and then extract the first blob file, wait to read the blob file and bind it to a global variable,

When the file reader loads bind the result of file reader to a var result, split that result by ‘BREAK’ and then resolve with the split result

**CODE THE ROUTINE**

class TextReader {

getFile (file) {

return new Promise(function (resolve) {

let reader = new FileReader()

reader.onload = function found () {

let result = reader.result

let resultSplit = []

resultSplit = result.split('BREAK')

resolve(resultSplit)

}

reader.readAsText(file)

})

}

async findGameLevels (event) {

let file = event.target.files[0]

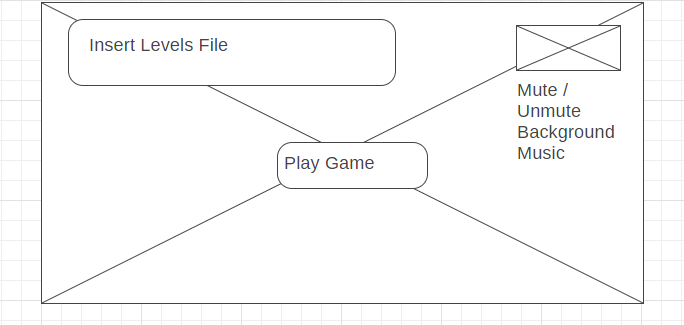
let result = await this.getFile(file)

window.allMyLevels = result

**CHECK THE CODE**

Working Great!!!

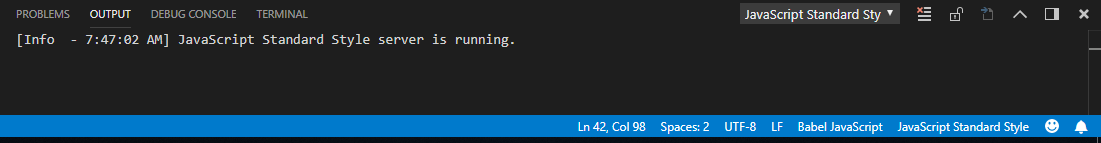
A plan for how the program feature you are working on will work [UML dynamic diagram, story-boards, wireframe, pseudocode]:



The Background X is animated, also the levels file is a text file now, so a user can edit it easily.

POST CODE COMPLETION>>>>>

A report showing nil style defects in your code according to JavaScript Standard Style https://standardjs.com/index.html:



Mistakes were made! A description and analysis of the mistakes made in the iteration:

Too many tasks in this iteration so I decided to stop the iteration after doing the comment and split and the background animation since I spent over 5 hours on those sections, I didn’t understand callback methods but after extensive research I was able to implement a promise function in my code to read a text file.

Lessons were learned? A plan for doing ONE thing differently in the next iteration:

I want to do some more research into advanced JavaScript topics like promises.